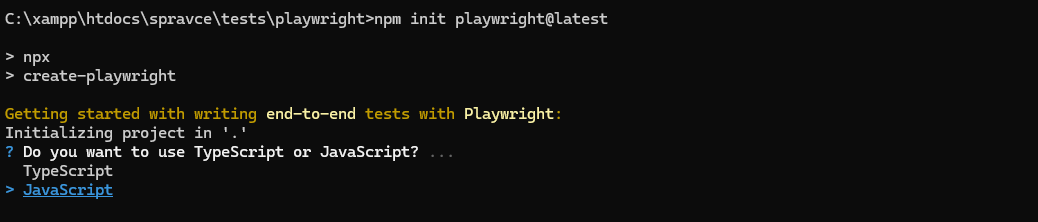
**Playwright**

**Inicializace projektu**

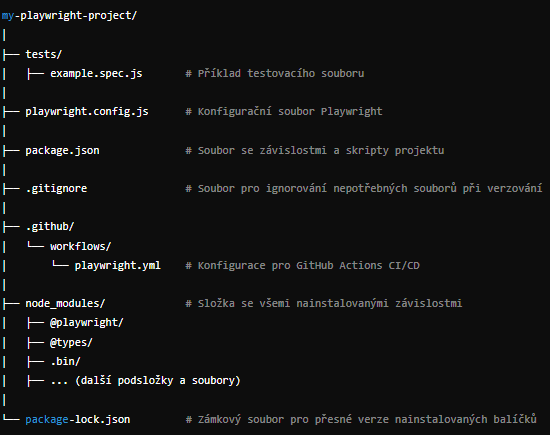
**npm init playwright@latest**

****

Příkazy - Playwright CLI – interakce s Playwright z příkazové řádky

|  |  |
| --- | --- |
| npx playwright test | Tento příkaz spustí všechny E2E testy, které jsou v projektu nadefinované. |
| npx playwright test –ui | Spustí interaktivní režim uživatelského rozhraní (UI), který umožnuje vizualizovat a spouštět testy, procházet výsledky. |
| npx playwright test –project=chromium | Tento příkaz spustí testy pouze v prohlížeči, který je definován jako chromium. |
| npx playwright test –debug | Spustí testy v debug režimu, kde mohu ladit testy s pauzami, vizualizovat akce a prohlížet si, co přesně testy dělají. |
| npx playwright codegen | Automaticky generuje testy pomocí Playwright Codegen, což je nástroj, který zaznamenává interakce s aplikací a generuje testovací kod. |
| npx playwright test <jmeno\_testu> | Spustí všechny testy, které jsou definované v jednom konkrétním souboru. |
| npx playwright show-report | Vygeneruje HTML test report. |

**Stromová struktura**



**Tests** – obsahuje testovací skripty.

**Playwright.config.js** – konfigurační soubor pro Playwright, kde mohu specifikovat různá nastavení, jako jsou prohlížeče, viewporty, globální konfigurace pro testy.

**Package.json** – obsahuje informace o projektu, včetně jeho názvu, verzí, zavislosti a skriptu, který mohu spustit pomocí npm.

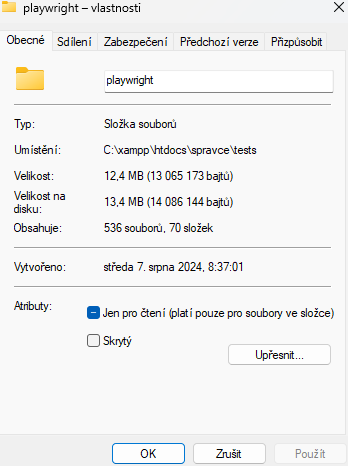
**.gitignore** – sem si mohu napsat všechny soubory a složky, které chci ignorovat při používání verzovacího systému GIT.

.**github/workflows/playwright.yml** – konfigurační soubor pro GitHub Actions, který nastavuje CI/CD process pro automatizované testování.

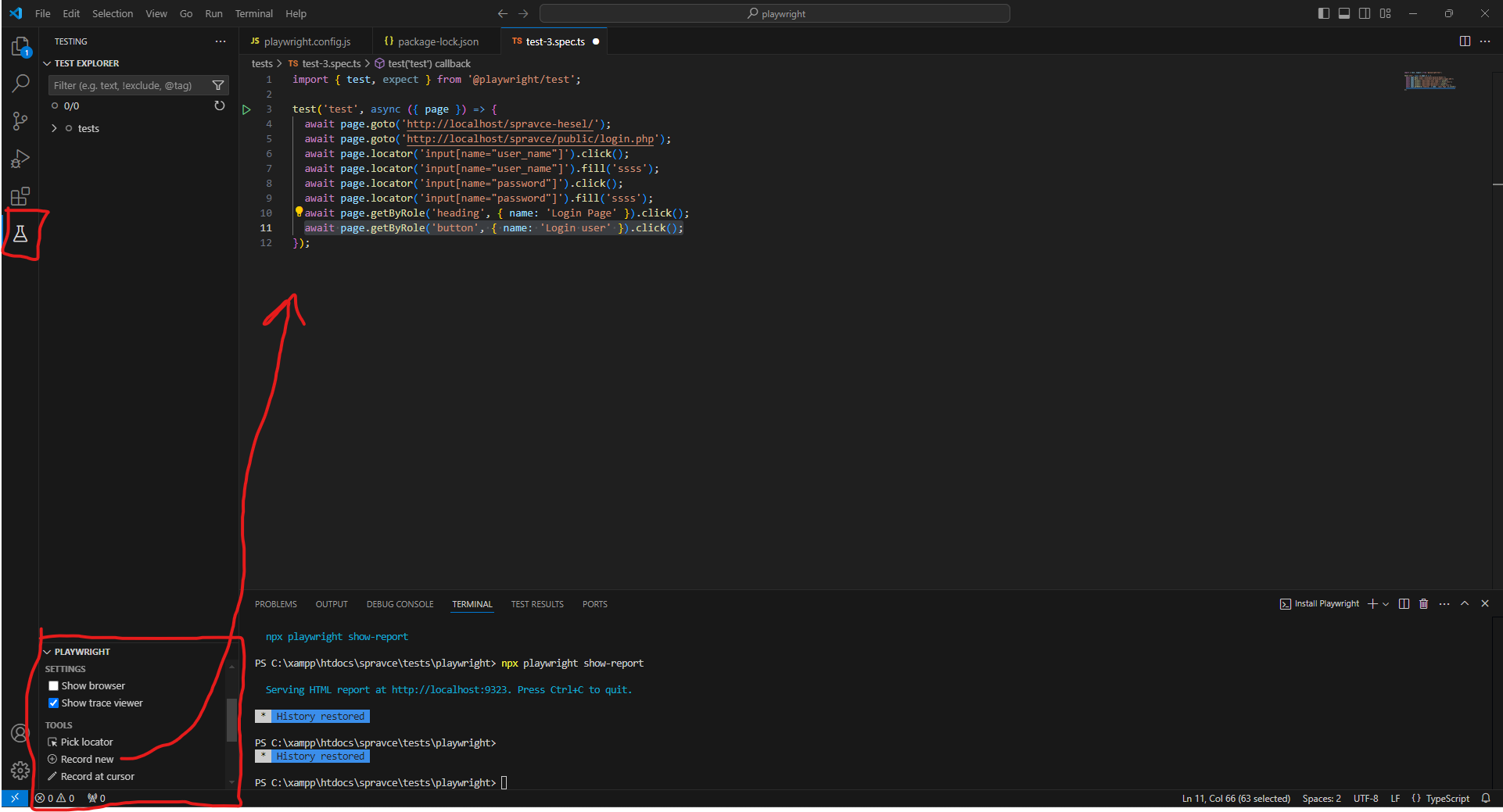
**node\_modules** – obsahuje všechny závislosti, které jsou nainstalovány pomocí npm podle specifikací v package.json (závislosti pro spuštění)

**package-lock.json** –zaznamenává přesné verze všech závislostí

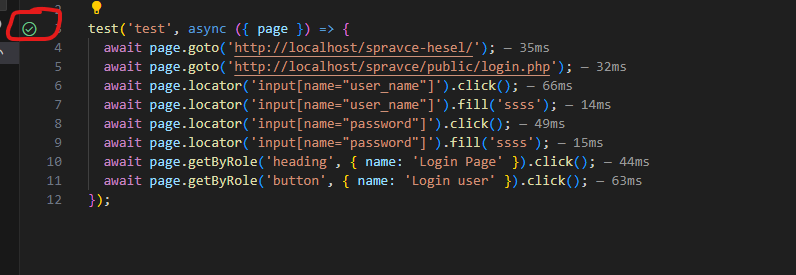
Pro zajímavost, velikost vygenerovaného prázdného projektu je 12MB.

**** + do gitu rozhodně není potřeba posílat složku node\_modules.

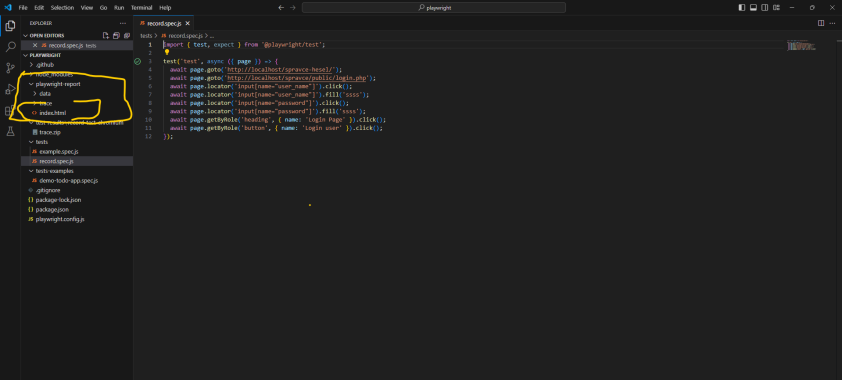
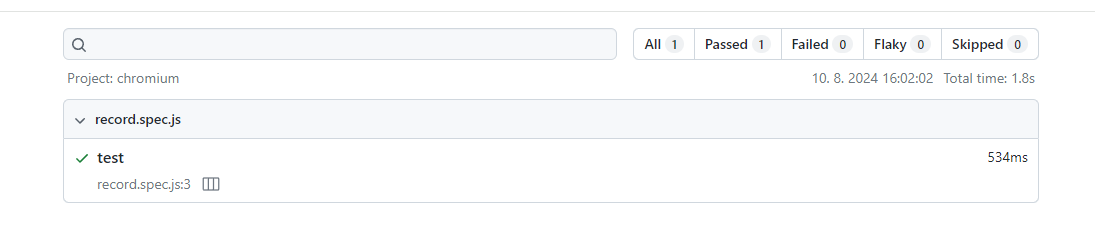
Přes vývojové studio – Visual studio code, dokážu naklikat můj scénář a VS mi na to vygeneruje zdrojový kod.



Jedno z možných spuštění testu ve VS.



Report z testu

Importy

Import {**test**,**expect**} from ‘@playwright/test‘;

**test**(“ověření titulku na stránce“,async({[page}) => {

tělo funkce

});

Importování funkce **test**, která je základem pro psaní jednotlivých testovacích případů. Každý test, který chci spustit, musí být definovaný pomocí teto funkce.

**Test** poskytuje způsob, jak definovat, co přesně má být v daném testu zkontrolováno.

**Expect** je funkce používaná k ověřování podmínek uvnitř testu.

Asynchronní funkce, označovaná v javascriptu klíčovým slovem **async**, je funkce, která umožnuje provádět asynchronní operace – operace, které mohou trvat nějaký čas a přitom neblokuje běh zpytku programu.

async function fetchData(){

try{

const response = await fetch(‘https://neco.cz‘); // čeká na odpověď z API

const data = await response.json(); // čeká na převod odpovědi do JSON formátu

return data; // vrací proměnou data -> promise

} catch (error){

console.error(‘Chyba při získávání dat:‘,error);

}

}

fetchData().then(data=>console.log(data)); // práce s daty po jejich načtení.

Promise – může mít tři stavy

1. Pending ( čekající)

* Asynchronní operace, kterou reprezentuje ještě nebyla dokončena. Promise je stále „čekající“ na výsledek

1. Fulfilled ( splněn )

* Asynchronní operace byla úspěšně dokončena a Promise je splněno, což je stav, který obsahuje hodnotu, která byla výsledkem operace – viz vzorovem příkladu, proměna data.

1. Rejected (odmítnut)

* Asynchronní operace selhala a nemohla úspěšně dokončit svůj úkol.

Důvody proč Promise mohl být odmítnut

1. Chyba na serveru
2. Siťové problémy
3. Neplatná URL
4. Problémy s převodem dat
5. Vyjímky v kodu – např: chyba v syntaxi, přístup k neexistujícímu objektu.

Přístupy k selektorům

V Playwright není potřeba explicitně uvádět typ selektoru jako je tomu v Selenium. Playwright je navržen tak, aby automaticky rozpoznal typ selektoru podle toho, jaký formát zadám.

|  |  |
| --- | --- |
| await page.click(‘input[**name**=”user\_name”]‘); | Výběr inputu na základě atributu **name** |
| await page.click(‘**#**submit-button‘) | Výběr elementu s **id** submit-button |
| await page.click(‘**.**login-form‘) | Výběr elementu s **třídou** login-form |
| await page.click(‘div > span‘) | Výběr spanu, který je přímý potomek divu |
| await page.click(‘xpath=//input[@name=”user\_name”]‘); | Výběr inputu na základě XPATH. |
| Await page.click(‘xpath=//div[@id=”login”]//button‘) | Výběr buttonu uvnitř divu s ID loginem. |

Přístup, který využívá Playwright k rozpoznávání selektorů dělá psaní testů jednodušší a zvyšuje jejich čitelnost.

|  |  |
| --- | --- |
| await page.getByRole() | K vyhledávání elementů na základě jejich role(button, link, atpd). |
| await page.getByText() | K vyhledání elementů na základě jejich textového obsahu. |
| await page.getByLabel() | K vyhledání ovládacího prvku formuláře podle souvisejícího textu štítku. |
| await page.getByPlaceholder() | Slouží k vyhledání vstupních polí na základě textu uvedeného v jejich placeholder. |
| await page.getByAltText() | Používá se k vyhledání prvků, obvykle obrazků, na základě jejich alternativního textu. |
| await page.getByTitle() | Slouží k vyhledání prvků na základě hodnoty jejich title atributu. |
| await page.getByTestId() | Používá se k vyhledání prvků na základě jejich data-testid |